pennsylvania
DEPARTMENT OF TRANSPORTATION

Cross-Asset Management Tools

FINAL REPORT

June 25, 2021

By Julie Drzymalski
Temple University

COMMONWEALTH OF PENNSYLVANIA
DEPARTMENT OF TRANSPORTATION

CONTRACT # 4400017651/511802
WORK ORDER # TEM WO 014

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FHWA-PA-2021-007-TEM WO 014		

4. Title and Subtitle	5. Report Date 6/24/21
Cross Asset Management Tools	6. Performing Organization Code
7. Author(s) **Julie Drzymalski**	**8. Performing Organization Report No.** 420977-5.2
9. Performing Organization Name and Address Temple University	**10. Work Unit No. (TRAIS)**
	11. Contract or Grant No. 4400017651/511802
12. Sponsoring Agency Name and Address The Pennsylvania Department of Transportation Bureau of Planning and Research Commonwealth Keystone Building 400 North Street, 6th Floor Harrisburg, PA 17120-0064	**13. Type of Report and Period Covered** 3/20/20 – 6/25/21
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

PennDOT seeks to find a methodology to prioritize work of their two largest asset classes, bridges and pavements using the Lowest Life Cycle Cost methodology (LLCC) to make more effective and efficient use of resources. Optimization of the projects is unrealistic given the volume of necessary work and complexity of the LLCC methodology. This research resulted in two heuristics which identify opportunities to combine projects into logical groups across asset classes. The results of each were compared and one which maximizes the amount of work grouped, was selected for further testing. Although the final logic was able to group 9.6% of work together, about 50% short of the amount expected, 20% of the work is completed within its recommended year and all work was completed within their acceptable window of delay (two years for pavements and five years for bridges). The results can be extended to incorporate additional asset classes or modified to maximize a different objective.

17. Key Words	18. Distribution Statement
Cross-asset optimization, lowest life cycle cost analysis,	No restrictions. This document is available from the National Technical Information Service, Springfield, VA 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	44	$42,044.00

Form DOT F 1700.7 **(8-72)** **Reproduction of completed page authorized**

This work was sponsored by the Pennsylvania Department of Transportation and the U.S. Department of Transportation, Federal Highway Administration. The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the Federal Highway Administration, U.S. Department of Transportation, or the Commonwealth of Pennsylvania at the time of publication. This report does not constitute a standard, specification, or regulation.

Contents

Executive Summary

Purpose

The purpose of this project was to enable the Pennsylvania Department of Transportation (PennDOT) to prioritize projects across asset classes (e.g., bridges and pavements), rather than only within each class, to make more effective and efficient use of resources and support PennDOT's commitment to managing assets for lowest life cycle cost (LLCC). To maximize efficiencies, PennDOT sought a tool that would identify opportunities to bundle individual "treatments" into logical groups ("project bundles") across asset classes.

Methodology

While there is an acceptable methodology for performing LLCC prioritization within an asset class, no methodology is in use for cross-asset allocation within the transportation sector. This is the subject of much current literature; a review is provided in Appendix B. This project, aiming to develop a usable tool for cross-asset allocation, comprised three parts:

1. The research team became familiar with PennDOT's current LLCC methodology, the logic and models utilized by PennDOT's Bridge Asset Management System (BAMS) and Pavement Asset Management System (PAMS), and factors the systems take into consideration.
2. The team developed and tested two approaches using a small data set supplied by PennDOT. The approaches utilized similar programming logic with different priority rankings. The team used SQL and MATLAB, exporting the program results to Excel. The data set was BAMS and PAMS output for PennDOT District 8 over the 12-year period of 2020–2031.
3. The programming logic was modified based on PennDOT feedback and a final approach was tested using a larger and more representative data set, also from District 8.

Results

This project successfully developed a prioritization tool that can develop a 12-year treatment schedule across two asset classes—bridges and pavements—while (a) maximizing the number of treatments in the same geographic area that can be bundled into projects and (b) minimizing the number of years that projects and treatments must be delayed to meet budget constraints.

Results highlights of the final prioritization tool:

- The final logic was able to bundle 9.6% of the treatments into projects, representing about 15 percent of the total 12-year costs.

- The average project bundle cost was $1.9 million, however the cost range was very large, from just over $100,000 to nearly $19 million.
- The tool was able to meet the established total bridge + pavement budget constraints for each of the 12 years. The bridge budget was exceeded in two of the years, however, because the original average bridge treatment/project cost exceeded the $75 million allowable budget, that constraint could not be met in every year.
- The final tool scheduled approximately 20% of the bridge or pavement treatments in their recommended year; all projects were scheduled within their acceptable windows of delay (within two years of recommended year for pavement work; within five years for bridge work).

PennDOT Implementation

The results of the District 8 pilot tests suggest that the cross-asset management tool could be applied to scheduling the 12-year program for any of PennDOT's Districts, and would be expected to:

- Strengthen management to LLCC by reducing deferred maintenance, despite the funding realities of constrained budgets;
- Enhance contracting and construction efficiency by bundling treatments for potential cost savings; and
- Balance costs for less fluctuation in total spending year-to-year.

The programming logic could be extended to incorporate additional asset classes, which would serve to increase the benefit of bundling treatments. With the two asset classes and the given data only 9.6% of the treatments were bundled; the addition of other assets should increase both the number of projects created as well as their size and scope. Further, depending on PennDOT priorities the logic could be changed to maximize the number of project bundles rather than minimizing treatment delays (the difference between actual treatment year and recommended year).

Introduction

The purpose of this project was to enable PennDOT to prioritize projects across asset classes (e.g., bridges and pavements), rather than only within each class, to make more effective and efficient use of resources and support PennDOT's commitment to managing assets for LLCC.

To maximize efficiencies, PennDOT sought a tool that would identify opportunities to bundle individual "treatments" (such as crack sealing a segment of pavement, milling and resurfacing a roadway, replacing a culvert, or rehabilitating a bridge) into logical groups to streamline contracting and construction. If work on multiple assets in the same geographic area can be scheduled concurrently, the resulting savings can enable more projects to be completed and support timely maintenance of assets. The grouped treatments are referred to as "project bundles" in this document. Note that the potential savings realized by bundling is not included in this project's calculations (because an exact value is extremely difficult to determine); future applications could include a reduction in treatment costs for projects that are bundled to further support LLCC prioritization.

This project developed, tested, and validated a project prioritization tool for PennDOT that determines a 12-year schedule across multiple asset classes within annual budget constraints, bundling the treatments where feasible. The asset classes tested were bridges and pavements.

Background

Currently, PennDOT has a sophisticated logic in its asset management software that determines optimal treatment schedules for each asset within an asset class. PennDOT's Bridge Asset Management System (BAMS) outputs a 12-year schedule for bridge treatments; the Pavement Asset Management System (PAMS) does the same for road treatments. The outputs are based upon current asset condition data and the expected future asset condition that would result from a specific treatment, or lack thereof. The logic aims to meet federal asset condition requirements and achieve LLCC of the assets. It follows a complex flowchart that uses a substantial data set of current conditions and models of future conditions given various inputs.

Due to budget constraints and other demands across Pennsylvania's extensive multimodal transportation system, the optimal bridge or pavement treatment often cannot be undertaken in the recommended year. The treatment is therefore shifted to a later year within a designated window—up to a two-year delay for pavement treatments and up to a five-year delay for bridge treatments. These intervals aim to provide reasonable flexibility for budgeting while limiting costlier deterioration of assets awaiting maintenance.

This project was undertaken to develop a methodology to generate a combined 12-year schedule across two asset classes—bridges and pavements—rather than within each class independently, bundling treatments into projects where possible. The project comprised three parts. First, the research team became familiar with PennDOT's current LLCC methodology, the logic and models utilized by BAMS and PAMS, and factors the systems take into consideration. Next, the team developed and tested two approaches using a small data set from PennDOT District 8, supplied by PennDOT. The approaches utilized similar logic with two different priority rankings. The third step was to modify the logic based on PennDOT feedback and test a final approach using a larger and more realistic data set.

There is limited research within the transportation industry on methodologies for cross-asset allocation using either optimization or a heuristic—the field is still in its infancy (see Appendix B for a literature review). Note that while true optimization across these two asset classes would be ideal, it is impractical given the volume of treatments and possible scenarios that would result. Because bridge (pavement) treatments can be performed within five (two) years of the ideal treatment year, the solution space or number of possible schedules is too vast to optimize. Thus, two heuristics (practical, logic-based approaches) were used and compared for proof of concept.

Phase One: Proof of Concept

The data supplied contained approximately 4,750 recommended bridge and pavement treatments for PennDOT District 8 over the years 2020–2031. Approximately one-third of the treatments had values equal to or less than $5,000. After discussions, PennDOT agreed that those smaller projects would be removed from the data set, leaving about 3,210 treatments to be scheduled to best achieve LLCC within a specified set of constraints.

Methodology

To the extent possible, treatments were to be bundled to create projects. The treatments and/or projects could be shifted to later years, within specified limits, to ensure the yearly budget constraints were met. This was done using the following constraints and assumptions:

1. Pavement treatments must be completed within two years of recommended date.
2. Bridge treatments must be completed within five years of recommended date.
3. Yearly bridge budget should not exceed $75 million.
4. Yearly pavement budget should not exceed $175 million.
5. Assume no committed projects (carryover projects from previous construction programs).
6. Only treatments with identical location IDs (indicating same county, route, and segment number) may be grouped into projects.
7. Costs for project bundles are allocated according to the following schedule:
   - $0-30M paid in same year
   - $30-50M paid over two years
   - $50-75M paid over three years

- $75-100M paid over four years
- >$100M paid over five years

Two approaches with different priorities were developed for the purposes of output comparison. Each approach created projects by bundling treatments in the same location and each approach shifted treatments and projects as needed to meet budget constraints. Logic 1 prioritizes the budget first, then creates as many projects from those treatments as possible in a given year, while Logic 2 prioritizes bundled project creation and then meets budget constraints by shifting timeframes. An overview of the logic follows. Further detail on the logic used and references to the corresponding Excel files are provided in Appendix A.

Logic 1: In this approach, the budget constraints are met first by shifting the treatments as needed, and then as many projects as possible are created within each year without further shifting of the schedule. The overall steps were:

1. Clean data so that treatments costing <$5,000 and blank rows are removed.
2. Determine location by concatenating (joining) codes for county, route, and segment number to generate an ID code indicating location. Differentiate pavement projects with a preceding "R." Combine BAMS and PAMS data sets.
3. Utilize MATLAB[1] program to shift projects in the following sequence:
    a. If the sum of recommended pavement treatments exceeds the $175M budget in any year, meet the budget by shifting treatments to later years in descending monetary order (most expensive treatment is shifted first). Pavement treatments may be delayed a maximum of two years.
    b. If the sum of recommended bridge treatments exceeds the $75M budget in any year, meet the budget by shifting treatments to later years in descending monetary order. Bridge treatments may be delayed a maximum of five years.
4. Utilize MATLAB to search for treatments scheduled for the same year and sharing the same location that can be bundled into projects.

Logic 2: Project creation is prioritized by identifying all possible project bundles, keeping in mind the maximum number of years a bridge or pavement project can be delayed. In each project, the year of the treatment which minimizes the total movement of other treatments will be set as the year in which to perform the bundled project. Thus, one treatment in each project serves as an "anchor" and will never be shifted to a different year. If budget constraints are not met with those dates, the bundled projects are shifted accordingly. Remaining individual treatments are then scheduled, and budget constraints are met by shifting individual treatments in descending monetary order. The overall steps were:

1. Clean data so that treatments costing <$5,000 and blank rows are removed.

[1] A mathematical programming environment. https://www.mathworks.com/products/matlab.html

2. Concatenate codes for county, route, and segment number to generate ID code indicating location. Differentiate pavement projects with a preceding "R." Combine BAMS and PAMS data sets.
3. Utilize MATLAB to search for treatments with the same location and within an appropriate year range (+/- 2 years for pavements; +/- 5 years for bridges) that can be bundled into projects, identifying how many years the treatments will need to be shifted to form bundled projects. Initially, at least one treatment in each bundle will not be shifted and acts as an anchor for the other treatments forming that project bundle. If budget constraints are not met, move projects in descending monetary order (up to the maximum allowable time frame (two or five years).
4. Shift remaining individual treatments in the following sequence to meet the budget:
    a. If the sum of recommended pavement treatments and projects exceeds the $175M budget in any given year, meet the budget by shifting treatments to later years in descending monetary order. Pavement treatments may be delayed a maximum of two years.
    b. If the sum of combined pavement and bridge work exceeds the yearly total budget of $250M in any year, shift bridge treatments in descending monetary order (up to a maximum of five years) to meet the budget.

Results

Project Creation and Budget Performance

The two methodologies provided different results in terms of the number of projects created. Table 1 highlights these differences.

Table 1: Bundled Project Comparison – Phase One

	Logic 1	Logic 2
Number of project bundles	52	60
Total project cost	$70,316,056	$68,905,989
Minimum project cost	$3,000	$11,000
Maximum project cost	$18,296,000	$18,296,000

It should be noted that no projects contained more than two treatments.

Table 2 details the original costs per year from the BAMS and PAMS output; years in which the budget is exceeded (greater than $75 million for bridges and $175 million for pavements) are highlighted in red. Table 3 is the result after each of the methodologies were employed.

Table2: Budget Adherence: Original Data

	Sum of BAMS-Recommended Treatments ($)	Sum of PAMS-Recommended Treatments ($)	TOTAL ($)
2020	69,224,000	425,335,357	494,559,357
2021	85,770,000	39,076,665	124,846,665
2022	86,787,000	9,826,440	96,613,440
2023	102,108,000	13,920,555	116,028,555
2024	102,907,000	14,895,467	117,802,467
2025	103,182,000	17,890,875	121,072,875
2026	103,342,000	27,048,358	130,390,358
2027	94,748,000	25,669,324	120,417,324
2028	79,840,000	458,001,725	537,841,725
2029	87,318,000	56,239,371	143,557,371
2030	88,288,000	23,895,747	112,183,747
2031	77,601,000	43,843,760	121,444,760

Table 3: Budget Adherence – Phase One Small Data Sample

	Logic 1			Logic 2		
Year	Bridge Cost ($)	Pavement Cost ($)	Total Cost ($)	Bridge Cost ($)	Pavement Cost ($)	Total Cost ($)
2020	69,224,000	174,311,925	243,535,925	64,845,000	174,718,154	239,563,154
2021	64,734,000	173,071,661	237,805,661	69,365,000	173,345,241	242,710,241
2022	25,391,000	118,889,758	144,280,758	22,347,000	118,046,810	140,393,810
2023	79,767,000	13,670,347	93,437,347	78,884,000	14,178,344	93,062,344
2024	97,618,000	14,252,179	111,870,179	92,189,000	13,394,473	105,583,473
2025	52,652,000	16,882,390	69,534,390	46,754,000	17,224,178	63,978,178
2026	98,102,000	25,523,517	123,625,517	95,663,000	26,425,360	22,088,360
2027	55,021,000	24,616,339	79,637,339	70,783,000	26,072,419	96,855,419
2028	72,997,000	174,783,143	247,780,143	72,835,000	174,517,497	247,352,497
2029	68,803,000	174,366,071	243,169,071	63,398,000	174,098,206	237,496,206

2030	56,665,000	174,470,713	231,135,713	65,833,000	174,125,663	239,958,663
2031	66,526,000	48,363,587	114,889,587	83,616,000	49,242,148	132,858,148

Logic 1 and Logic 2 both succeeded in creating more balanced costs year-to-year, lowering the standard deviation of yearly total costs from $155.7M to $69.3M (Logic 1) and $68.5M (Logic 2). Because the original average bridge treatment/project cost was $90M per year, there was no way of keeping each year within its $75M allowable budget. Both methods had the negative effect of increasing the deviation on the bridge budgets from $10M in the original to $18.9M.

Schedule Performance

Both logics succeeded in completing all treatments within their allowable limits for optimal life cycle (within two years of recommended year for pavements; within five years for bridges). The number and percentage of treatments that were shifted to a later year are shown in Table 4. Logic 1 shows slightly better results: 90% of all treatments would be performed in their recommended year, with just under 10% of the treatments needing to be shifted to a later year.

Table 4: Quantity of Treatment Shifts by Number of Years Shifted

	Number of Years Shifted					Total
	1	2	3	4	5	
Logic 1						
Number of Treatments Shifted	201	260	0	0	0	**461**
Percentage of Treatments Shifted	4.26%	5.52%	0.00%	0.00%	0.00%	**9.78%**
Logic 2						
Number of Treatments Shifted	272	307	26	21	16	**642**
Percentage of Treatments Shifted	5.77%	6.51%	0.55%	0.45%	0.34%	**13.62%**

Phase Two: Prototype Testing

The PAMS data set supplied for Phase Two was modified from the Phase One data set with updated PennDOT LLCC logic. After discussions with PennDOT, only those treatments exceeding $50,000 were to be considered for both pavement and bridges, leaving a similarly sized data set as in Phase One to be scheduled to best achieve LLCC within a specified set of constraints.

Methodology

PennDOT wanted to prioritize project bundle creation, thus only Logic 2 was utilized in this phase because it bundles the projects first and prioritizes their completion. The desired approach was to maximize the number of project bundles (PennDOT's target was around 300) while minimizing the number of years a treatment is delayed from its recommended year. This is essentially what Logic 2 accomplishes; it creates all possible project bundles first, then shifts them across years as needed. Logic 1 is constrained as to how many projects can be created because the budget constraints are met first, therefore no further shifting can occur.

The methodology was modified in the following ways:

1. Data was cleared of any projects under $50,000 (changed from the $5,000 threshold used for Phase One).
2. After pavement project bundles were scheduled, and individual pavement treatments were moved to meet the $175M budget, the bridge treatments were moved to meet the $75M bridge budget, rather than to meet an overall budget of $250M as was used in Phase One. The Phase One approach allowed any excess pavement funding in any given year to be applied to bridge needs, which does not reflect actual funding policy.

The same constraints were employed for the Phase Two test as were used in the Phase One proof-of-concept test:

1. Pavement treatments must be completed within two years of recommended date.
2. Bridge treatments must be performed within five years of recommended date.
3. Yearly bridge budget should not exceed ~$75M.
4. Yearly pavement budget should not exceed ~$175M.
5. Assume no committed projects.
6. Only treatments with identical location IDs can be grouped into projects (indicating same county, route, and segment number).
7. Costs for bundled projects are allocated according to the following schedule:
   - $0-30M paid in same year
   - $30-50M paid over two years
   - $50-75M paid over three years
   - $75-100M paid over four years
   - >$100M paid over five years

Results

Project Creation and Budget Performance

In Phase Two, 9.6% of the treatments were able to be bundled into projects. Specifically, the output yielded:

- 150 project bundles comprising 294 treatments, or about 10% of all treatments
- Total project cost = $288,588,277, or about 15% of the total 12-year cost of all projects and treatments
- Average project cost = $1,923,922
- Minimum project cost = $123,693
- Maximum project cost = $18,296,000

The number of bundled projects (150) was significantly less than the desired value of approximately 300 projects. Further, only seven of the 150 projects contain more than two treatments. The spread of project costs is also something to consider as they range from just over $100,000 to nearly $19,000,000.

Table 5 details the count of projects and remaining individual treatments each year; Table 6 details the costs of those projects and displays over-budget years in red. This data set and new logic yielded more than double the number of project bundles that were able to be created and scheduled. Additionally, the budgets are more balanced with only two bridge treatment budgets exceeding $75M.

Table 5: Phase Two Number of Project Bundles and Individual Treatments by Year

Year	Project Bundles	Individual Bridge Treatments	Individual Pavement Treatments
2020	10	96	260
2021	9	74	174
2022	6	15	102
2023	15	72	198
2024	11	58	168
2025	16	52	189
2026	9	61	149
2027	12	58	147
2028	15	51	206
2029	11	46	198
2030	18	53	255
2031	18	53	378
TOTAL	**150**	**689**	**2,424**

Table 6: Budget Adherence – Phase Two Prototype Testing

	Projects			Individual Treatments			Total		
	Bridge Cost ($)	Pavement Cost ($)	Total Cost ($)	Bridge Cost ($)	Pavement Cost ($)	Total Cost ($)	Bridge Cost ($)	Pavement Cost ($)	Total Cost ($)
2020	9,399,000	2,806,489	12,205,489	53,802,000	108,935,732	162,737,732	63,201,000	111,742,221	174,943,221
2021	5,631,000	2,784,471	8,415,471	52,817,000	75,955,893	128,772,893	58,448,000	78,740,364	137,188,364
2022	13,769,000	3,679,134	17,448,134	9,439,000	44,866,149	54,305,149	23,208,000	48,545,283	71,753,283
2023	23,400,000	5,621,637	29,021,637	44,286,000	95,014,638	139,300,638	67,686,000	100,636,275	168,322,275
2024	30,782,000	4,767,454	35,549,454	56,264,000	93,432,184	149,696,184	87,046,000	98,199,638	185,245,638
2025	24,589,000	5,912,031	30,501,031	37,289,000	90,576,498	127,865,498	61,878,000	96,488,529	158,366,529
2026	14,584,000	3,821,978	18,405,978	76,490,000	96,864,174	173,354,174	91,074,000	100,686,152	191,760,152
2027	21,417,000	6,068,647	27,485,647	42,332,000	91,114,954	133,446,954	63,749,000	97,183,601	160,932,601
2028	19,993,000	7,783,643	27,776,643	48,319,000	102,829,364	151,148,364	68,312,000	110,613,007	178,925,007
2029	15,044,000	3,988,836	19,032,836	52,894,000	96,132,238	149,026,238	67,938,000	100,121,074	168,059,074
2030	30,392,000	8,893,662	39,285,662	44,825,000	104,887,030	149,712,030	75,217,000	113,780,692	188,997,692
2031	16,843,000	6,617,295	23,460,295	58,419,000	93,458,116	151,877,116	75,262,000	100,075,411	175,337,411

Schedule Performance

Of particular interest is the number of years each treatment needed to be shifted beyond its recommended year, because the longer the delay, the greater the potential deterioration of the asset, which is counter to LLCC principles. The quantity of projects that had to be delayed, by number of years delayed (up to two for pavements, five for bridges), is shown in Table 7. Half (56) of the treatments shown in Table 6 were shifted to accommodate a project bundling rather than due to budget constraints.

Table 7: Number of Projects Shifted by Number of Years Delayed

0 years	1 Year	2 Years	3 Years	4 Years	5 Years
28	33	28	23	14	14

This is an improvement from the Phase One testing which had only about 10% being completed in the recommended year, however the improvement may be due to the randomness of the data or improved data accuracy in the PAMS outputs used for Phase Two, rather than the enhanced logic.

Conclusion

This project successfully developed a prioritization tool that can develop a 12-year treatment schedule across two asset classes—bridges and pavements—while (a) maximizing the number of treatments in the same geographic area that can be bundled into projects and (b) minimizing the number of years that projects and treatments must be delayed to meet budget constraints.

The tool uses a heuristic—a logic-based system that produces practical results—rather than an optimization, which in theory produces the perfectly ideal solution. The logic in the final heuristic prioritizes project bundling by first finding the set of all possible treatments that can be bundled into projects based on location data and then shifting those projects and the remaining treatments to later years if needed to meet budget constraints (up to two years later for pavements and up to five years later for bridges). Given the size of the data set and the number of possible scenarios that are created by the two- and five-year windows, it is unlikely that the tool produces the truly optimal solution but rather a near-optimal solution, which is nevertheless useful for PennDOT's purposes.

The tool was tested using the list of recommended bridge and pavement treatments for PennDOT's District 8 for the 12-year period of 2020–2031, as generated by PennDOT's BAMS and PAMS systems. The results of the testing suggest that the tool could be applied to scheduling the 12-year program for any of PennDOT's Districts, and would be expected to:

- Strengthen management to LLCC by reducing deferred maintenance despite the funding realities of constrained budgets;
- Enhance contracting and construction efficiency by bundling treatments for potential cost savings; and
- Balance costs for less fluctuation in total spending year-to-year.

While the logic succeeded in better balancing costs year-to-year, the total spending per year still fluctuated greatly—by approximately $28M for an overall budget of $250M. Additionally, any excess in the individual asset budgets in a particular year went unused by the other asset. Although project bundles were created, the number of bundles generated was less than desired (about half PennDOT's target number). While this could be data-set-related, it does diminish the benefit of bundling.

Opportunities for future tool enhancement could include expanding the logic and parameters to accommodate additional asset classes, which would also increase bundling opportunities. In addition, adjustments could be made to the logic to potentially increase the number of project bundles. After the set of possible bundled projects is identified, instead of selecting projects based on minimizing the number of years that treatments must be shifted, the number of project bundles that are created in the final solution could be maximized. This could potentially require an iterative solution, if done heuristically, which could take significant computing effort. Greater benefits could also be seen by allocating unused funds from one asset in a particular year to the other asset.

Finally, it should be noted that while the tool appears to allocate transportation spending efficiently, in the increasingly common scenario where available funding cannot meet the asset management needs of the transportation system, the value of the tool would be diminished.

Appendix A: User Implementation Guide

PennDOT Bridge/Pavement Optimization Tool

User Implementation Guide

June 2021

Overview

This guide explains the steps used to create a final combined schedule of bridge and pavement jobs and projects between 2020 and 2031 while operating under established budgetary constraints. For the purposes of this guide, the term "job" refers to an individual pavement or bridge treatment, whereas the term "project" refers to a package of jobs (a project bundle) based on a common location.

This guide describes the two following methods used to obtain a final schedule:

- The first process prioritizes yearly budget constraints by shifting single jobs first to meet budget, then packages projects within the same year based on location.
- The second process prioritizes packaging projects based on location, then shifts single jobs to meet the budget.

Getting Started

In order to run these two optimization programs, the user will need the following installed on their computer:

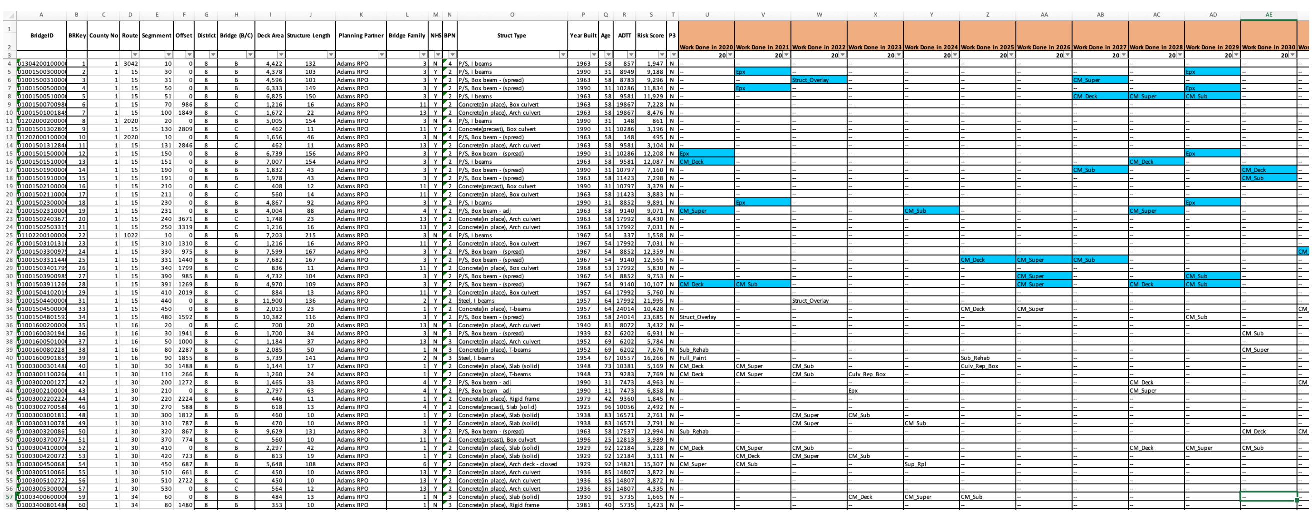
1. Microsoft Excel (for loading and viewing PennDOT inputs and results)
2. A SQL workbench and server linked to Excel (for initial data sorting and extraction). The system used in this guide is the open-source MySQL Workbench.
3. MATLAB (for running the optimization programs)

Process 1 (Prioritizing Budget First)

As discussed in the overview, the logic of the first process is to shift single jobs first to meet budgetary constraints by year, then identify and package jobs into projects within each year based on the schedule output.

1. Standardizing Initial Excel Data

The data received from PennDOT comes in the form of two individual BAMS (bridges) and PAMS (pavements) reports that must be standardized prior to data analysis. These reports are Excel spreadsheet files that account for a number of durability and condition factors to forecast maintenance needs across a 12-year horizon. The reports show maintenance type and estimated cost at each location. The image below is an example of the first few columns of an initial BAMS report. Scrolling to the right on the spreadsheet reveals projected cost in each year.



The relevant columns for this optimization process are BridgeID (A), BRKey (B), the 11 “Work Done in...” columns (U-AG), and “Cost” columns for each year (BV, CM, etc.). All other columns are manually removed, leaving a simplified spreadsheet that is compatible with SQL and MATLAB.

BRKey	BridgeID	Work Done in 2020	Work Done in 2021	Work Done in 2022	Work Done in 2023	Work Done in 2024	Work Done in 2025	Work Done in 2026	Work Done in 2027	Work Done in 2028	Work Done in 2029	Work Done in 2030	Work Done in 2031
1	01304200100000												
2	01001500300000		Epx								Epx		
3	01001500310000			Struct_Overlay					CM_Super				
4	01001500500000		Epx								Epx		
5	01001500510000								CM_Deck	CM_Super	CM_Sub		
6	01001500700986												
7	01001501001849												
8	01202000200000												
9	01001501302809												
10	01202000100000												
11	01001501312846												
12	01001501500000	Epx									Epx		
13	01001501510000	CM_Deck								CM_Deck			
14	01001501900000								CM_Sub			CM_Deck	
15	01001501910000											CM_Sub	
16	01001502100000												
17	01001502110000												
18	01001502300000		Epx								Epx		
19	01001502310000	CM_Super				CM_Sub				CM_Super			
20	01001502403671												
21	01001502503319												
22	01102200100000												
23	01001503101310												
24	01001503300975												CM_Super
25	01001503311440						CM_Deck	CM_Super	CM_Sub				
26	01001503401799												
27	01001503900985							CM_Super			CM_Sub		
28	01001503911269	CM_Deck	CM_Sub					CM_Super		CM_Deck	CM_Sub		
29	01001504102019												
31	01001504400000			Struct_Overlay									
33	01001504500000						CM_Deck	CM_Super					CM_Sub
34	01001504801592	Struct_Overlay									CM_Sub		
35	01001600200000												
36	01001600301941											CM_Sub	
37	01001600501000												
38	01001600802287	Sub_Rehab										CM_Super	
39	01001600901855	Full_Paint					Sub_Rehab						
40	01003000301488	CM_Deck	CM_Super	CM_Sub			Culv_Rep_Box						
41	01003001100266	CM_Deck	CM_Super	CM_Sub	Culv_Rep_Box								
42	01003002001272									CM_Deck			CM_Super
43	01003002100000				Epx					CM_Super			
44	01003002202224												
46	01003002700588												
48	01003003001812			CM_Super	CM_Sub								
49	01003003100787			CM_Super		CM_Sub							
50	01003003200867	Sub_Rehab										CM_Deck	CM_Super

BRKey	Cost 2020	Cost 2021	Cost 2022	Cost 2023	Cost 2024	Cost 2025	Cost 2026	Cost 2027	Cost 2028	Cost 2029	Cost 2030	Cost 2031
1												
2		113000								143000		
3			366000					6000				
4		163000								207000		
5								7000	9000	16000		
6												
7												
8												
9												
10												
11												
12	168000									220000		
13	6000								7000			
14								5000			3000	
15											6000	
16												
17												
18		125000								159000		
19	5000				8000				6000			
20												
21												
22												
23												
24												11000
25						7000	9000	16000				
26												
27							6000			11000		
28	4000	9000					7000		6000	12000		
29												
31			947000									
33						3000	3000					6000
34	779000									23000		
35												
36											5000	
37												
38	50000										4000	
39	201000					163000						
40	2000	2000	3000			862000						
41	2000	2000	3000	895000								
42									3000			3000
43				76000					4000			
44												
46												
48			2000	2000								
49			2000		2000							
50	131000										10000	13000

The same process is repeated with the PAMS report. However, the original PAMS report does not contain a Key or ID column, so those need to be manually inserted. The key in the PAMS spreadsheet is made by inserting the key "R1" into Row 1, then simply dragging and repeating down the rows. The "R" preceding the numerical value distinguishes pavement from bridges, as bridge keys are represented by a number with no preceding letter. The user also must manually add an ID column. The ID column is important because it represents the geographic location of each pavement section. In the BAMS report, the BridgeID was simply a concatenation of County Number, Route, Segment, and Offset into a 14-digit code:

01304200100000

County No: 1

Route: 3042

Segment: 10

Offset: 0

However, offset is not included in the PAMS report and therefore is not needed, so the last four digits from the BridgeID are removed to standardize it with the Pavement ID:

0130420010

An Excel concatenation of County Number, Route, and Segment in the PAMS report generates the Pavement ID, although it is important to remember the correct number of digit placeholders for each (two, four, and four, respectively). The BridgeID and Pavement ID in each spreadsheet are then renamed “TargetID” for standardization purposes between the two reports.

RDKey	TargetID	Work_Done_in_2020	Work_Done_in_2021	Work_Done_in_2022	Work_Done_in_2023	Work_Done_in_2024	Work_Done_in_2025	Work_Done_in_2026	Work_Done_in_2027	Work_Done_in_2028	Work_Done_in_2029	Work_Done_in_2030	Work_Done_in_2031	[redacted]	Cost_2020	Cost_2021	Cost_2022	Cost_2023	Cost_2024	Cost_2025	Cost_
R1	0100150010							Slab Replacement (JCP)													
R2	0100150011								Mechanized Patch (HMA&COMP)				Mechanized Patch (HMA								
R3	0100150020						Concrete Patching w/Dia	Slab Replacement (JCP)												400781	
R5	0100150030						Concrete Patching w/Dia	Slab Replacement (JCP)												398873	
R7	0100150040						Concrete Patching w/Dia	Slab Replacement (JCP)												419565	
R9	0100150050						Concrete Patching w/Dia	Slab Replacement (JCP)												347804	
R11	0100150060						Concrete Patching w/Dia	Slab Replacement (JCP)												422060	
R13	0100150070						Concrete Patching w/Dia	Slab Replacement (JCP)												360277	
R15	0100150080						Concrete Patching w/Dia	Slab Replacement (JCP)												385372	
R17	0100150090						Concrete Patching w/Dia	Slab Replacement (JCP)												571895	
R19	0100150100						Concrete Patching w/Dia	Slab Replacement (JCP)												444367	
R21	0100150110							Slab Replacement (JCP)													
R22	0100150111								Mechanized Patch (HMA&COMP)				Mechanized Patch (HMA								
R23	0100150120							Slab Replacement (JCP)													
R24	0100150121							Mechanized Patch (HMA&COMP)				Mechanized Patch (HMA&COMP)									
R25	0100150130							Slab Replacement (JCP)													
R26	0100150131								Mechanized Patch (HMA&COMP)			Mill, Mechanized Edge Pa	Mechanized Patch (HMA								
R27	0100150140								Slab Replacement (JCP)												
R28	0100150141										Mechanized Patch (HMA&COMP)										
R29	0100150150						Concrete Patch w/4” Overlay (JCP)													813643	
R30	0100150151							Mechanized Patch (HMA&COMP)		Mill, Mechanized Edge Patch (HMA&COMP)		Mechanized Patch (HMA&COMP)									
R31	0100150160							Slab Replacement (JCP)													
R32	0100150161							Mechanized Patch (HMA&COMP)		Mill, Mechanized Edge Patch (HMA&COMP)		Mechanized Patch (HMA&COMP)									
R33	0100150170							Slab Replacement (JCP)													
R34	0100150171							Mechanized Patch (HMA&COMP)		Mill, Mechanized Edge Patch (HMA&COMP)		Mechanized Patch (HMA&COMP)									
R35	0100150180							Slab Replacement (JCP)													
R36	0100150181								Mechanized Patch (HMA&COMP)		Mill, Mechanized Edge Patch (HMA&COMP)		Mechanized Patch (HMA								
R37	0100150190							Slab Replacement (JCP)													
R38	0100150191									Mechanized Patch (HMA&COMP)											
R39	0100150200							Concrete Patch w/4” Overlay (JCP)													
R40	0100150201									Mechanized Patch (HMA	Mill, Mechanized Edge Patch (HMA&COMP)										
R41	0100150210							Slab Replacement (JCP)													
R42	0100150211							Mechanized Patch (HMA&COMP)				Mechanized Patch (HMA&COMP)									
R43	0100150220								Slab Replacement (JCP)												
R44	0100150221									Mechanized Patch (HMA&COMP)											
R45	0100150230							Slab Replacement (JCP)													
R46	0100150231								Mechanized Patch (HMA&COMP)				Mechanized Patch (HMA								
R47	0100150240						Slab Replacement (JCP)	Concrete Patching w/Diamond Grinding (JCP)												1267	
R49	0100150250						Concrete Patching w/Dia	Slab Replacement (JCP)												517596	
R51	0100150260						Concrete Patching w/Dia	Slab Replacement (JCP)												338118	
R53	0100150270						Concrete Patching w/Dia	Slab Replacement (JCP)												564410	
R71	0100150360											Mill, Mechanized Patch (HMA&COMP)									
R74	0100150371												Mill, Mechanized Patch (		MP)						
R101	0100160010							Mill, Mechanized Edge Patch (HMA&COMP)		Base Repair, Manual Patch (HMA&COMP)											
R102	0100160020						Mill, Mechanized Edge Patch (HMA&COMP)		Base Repair, Manual Patch (HMA&COMP)											3781	
R103	0100160030						Mill, Mechanized Edge Patch (HMA&COMP)													2643	
R104	0100160040						Mill, Mechanized Edge Patch (HMA&COMP)			Base Repair, Manual Patch (HMA&COMP)										3063	
R105	0100160050					Mill, Mechanized Patch (HMA&COMP)				Mill, Mechanized Edge Patch (HMA&COMP)									9585		

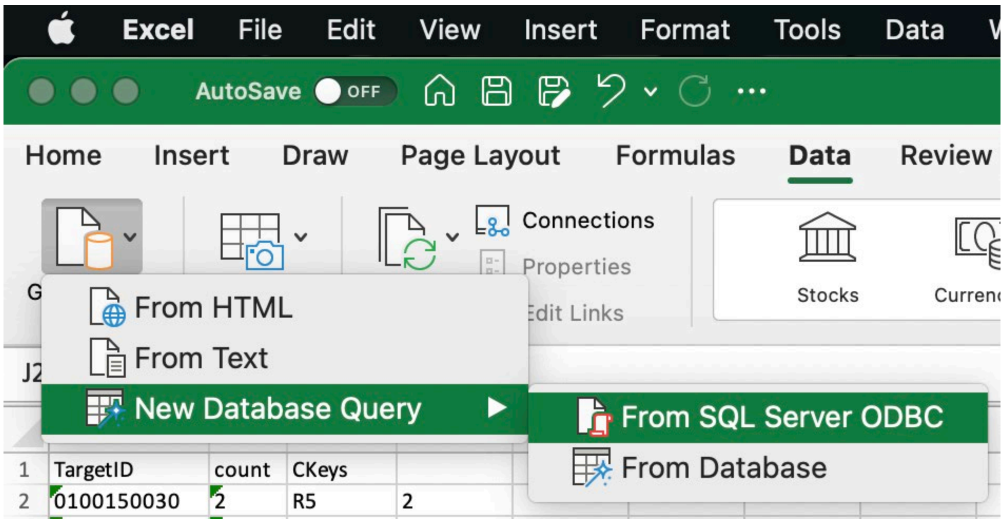
Now the PAMS and BAMS reports have been standardized and can be combined into one spreadsheet. The connecting point is shown below (notice the switch from Bridge to Pavement Keys highlighted by the preceding “R”).

1	CKey	TargetID	Work_Don	Work_Don	Work_Don	Work_Don	Work_Don	Work_Don	Work_Don	Work_Don	Work_Don	Work_Don	Work_Don	Work_Don	[redacted]	Cost_2020	Cost_2021	Cost_2022	Cost_2023	Cost_2024	Cost_2025	Cost_2026	Cost_2027	Cost_2028	Cost_2029	Cost_2030	Cost_2031
1504	55852	2800160530							Epx													136000					
1505	55909	6630530050											Epx													90000	
1506	56021	2809970680							Epx													103000					
1507	56115	6602160180					Epx													75000							
1508	56645	3610450120											Epx													109000	
1509	56737	6600740750					Epx													226000							
1510	57135	6602160260							Epx													57000					
1511	57235	0101160210							Epx													94000					
1512	R1	0100150010							Slab Replacement (JCP)													1623					
1513	R2	0100150011								Mechanized Patch (HMA&COMP)				Mechanize		A&COMP)							1212				1261
1514	R3	0100150020						Concrete P	Slab Replacement (JCP)												400781	1272					
1515	R5	0100150030						Concrete P	Slab Replacement (JCP)												398873	1266					
1516	R7	0100150040						Concrete P	Slab Replacement (JCP)												419565	1331					
1517	R9	0100150050						Concrete P	Slab Replacement (JCP)												347804	1104					
1518	R11	0100150060						Concrete P	Slab Replacement (JCP)												422060	1339					
1519	R13	0100150070						Concrete P	Slab Replacement (JCP)												360277	1143					
1520	R15	0100150080						Concrete P	Slab Replacement (JCP)												385372	1223					
1521	R17	0100150090						Concrete P	Slab Replacement (JCP)												571895	1815					

2. Using SQL Scripts to Remove Extraneous Data

In this situation, SQL is a useful tool for data manipulation prior to importing into MATLAB. The first step is to remove extraneous rows where there are no jobs scheduled. There were several instances where certain bridge or pavement segments existed with no scheduled maintenance. Using SQL those rows are removed from the Excel spreadsheet to decrease run time and simplify the data.

In order to run SQL scripts on an Excel spreadsheet, the user must install a SQL workbench and a server to establish a connection between the workbench and Excel. The development team found MySQL (free open-source software) to be the easiest platform to use. When the proper software is installed, the user establishes a connection between the workbench and Excel via the Data tab in Excel.



After this connection is established, the user can run scripts in MySQL Workbench and then import the results back into Excel. In this manual, the master spreadsheet created in the previous section will be referred to as *combined_data*.

The first step is to run a script that removes the extraneous rows with no jobs. This can be performed with a command like shown below:

```
DELETE from combined_data where Work_Done_in_2020 is null and
Work_Done_in_2021 is null and Work_Done_in_2023 is null and Work_Done_in_2024 is
null and Work_Done_in_2025 is null and Work_Done_in_2026 is null and
Work_Done_in_2027 is null and Work_Done_in_2028 is null and Work_Done_in_2029 is
null and Work_Done_in_2029 is null and Work_Done_in_2030 is null and
Work_Done_in_2031 is null;
```

Another script can be used at the user's discretion to remove projects under a certain cost. In this example of code, all projects less than $5,000 are removed.

```
update combined_data

set Work_Done_in_2020 = Null
```

```
where Cost_2020 < 5000;


update new_road_data

set Work_Done_in_2021 = Null

where Cost_2021 < 5000;
```

...and so on until the last year

3. Using SQL Scripts to Find Potential Pairings

The final step with SQL prior to importing the database into MATLAB is to find bridges and pavement segments that share a common TargetID, which identifies the geographic location as explained earlier. The SQL script below groups the keys of bridges and pavement sections into geographic groups:

```
SELECT TargetID, count(*) as count, group_concat(CKey Separator', ' ) as CKeys FROM
combined_data

group by TargetID

having count(*) > 1

order by 1;
```

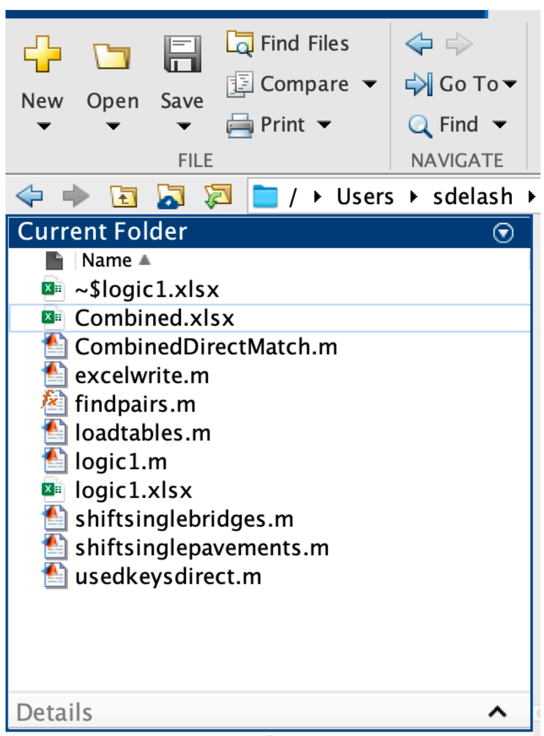
This command creates a table of all the possible geographic groupings. This table is added onto another sheet in the *combined_data* master spreadsheet and is now ready for importation.

	A	B	C	D	E	F
1	TargetID	count	CKeys			
2	0100150030	2	R5	2		
3	0100150050	2	4	R9		
4	0100150150	2	R29	12		
5	0100150151	2	R30	13		
6	0100150190	2	14	R37		
7	0100150191	2	R38	15		
8	0100150230	2	18	R45		
9	0100150231	2	19	R46		
10	0100160030	2	36	R103		
11	0100300110	2	R122	41		
12	0100300200	2	R131	42		
13	0100300300	2	48	R143		
14	0100300310	2	R144	49		
15	0100300320	2	R145	50		
16	0100300410	2	52	R156		
17	0100300420	2	R157	53		
18	0100300450	2	R160	54		
19	0100340060	2	R182	59		
20	0100340110	3	61	R187	63	
21	0100340120	2	64	R188		
22	0100340190	2	69	R195		
23	0100340240	2	R200	72		
24	0100340260	2	74	R202		
25	0100940120	2	76	R217		
26	0100940130	2	77	R218		
27	0100940240	2	R229	80		
28	0100970010	3	85	R242	84	
29	0100970040	2	86	R245		
30	0100970120	2	R253	88		
31	0101160010	2	93	R262		

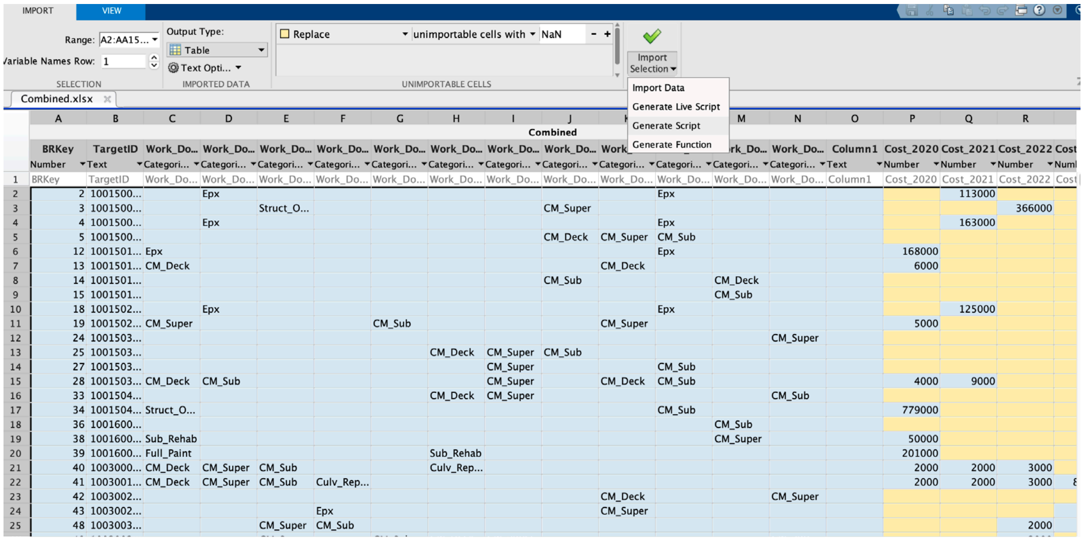
bridges | roads | combined_data | possible

4. Loading the Spreadsheet into MATLAB

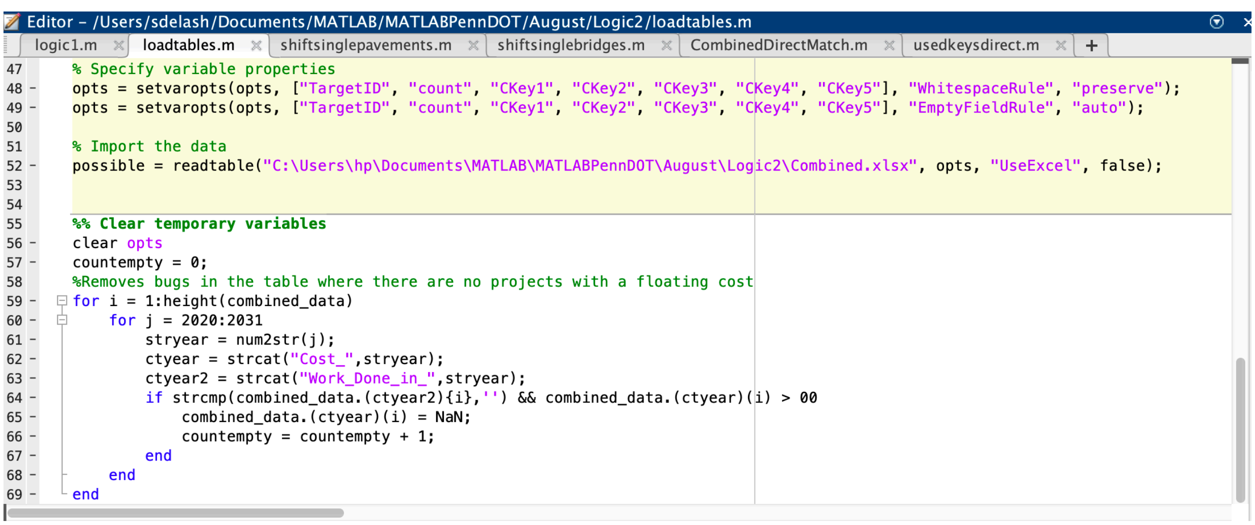
Now, the master data set (named "Combined.xlsx" in these screenshots) is ready to be imported into MATLAB for optimization. The first step is to save the file to the current MATLAB path folder where all the .m file scripts will be saved. The spreadsheet will appear in this window when saved to the path.



Double-clicking the spreadsheet opens the window below, where the user selects "Generate Script," creating a program that loads the spreadsheet into MATLAB.



Now MATLAB will autogenerate a script to load the spreadsheet into MATLAB. The user can then add manual tweaks below this autogenerated script based on preferences for the data that weren't addressed in SQL. For example, the block of code below removes random floating costs with no associated job that may have been a bug in the original PAMS and BAMS data set.



5. Shifting Single Jobs

Now that loading is complete, the first step is move single pavement jobs to meet the given pavement budgetary constraint of $175M per year. The script *shiftsinglepavements.m* is the MATLAB file that executes this process. Pavement jobs can be delayed up to two years.

Next, single bridge jobs are shifted to meet the total budget of $250M per year. The script *shiftsinglebridges.m* is the MATLAB file that executes this process. Bridge jobs can be delayed up to five years.

Detailed notes on the code can be found in the comments of the code, but the general logic of these two scripts is described below.

The program first sums the costs of all the pavement jobs for each column (year), given from row 1511 to the bottom of the table. The 1511 would need to be amended if the pavement jobs began in a different row.

```
for i = 1:12
    rsum(i) = nansum(combined_data.(i+15)(1511:height(combined_data)));
end
```

The program then scans the first column to see if the costs are over budget. If they are, it checks if the next year's costs are over budget as well to ensure that a project can be delayed to the next year without exceeding the next year's budget. If the current column is over budget and the next column is under budget, then the most expensive project in the current year is moved one year and stamped with one asterisk, indicating it has been moved one year. The program does not move jobs already stamped with two asterisks, as pavement jobs can only be shifted up to two years.

```
while not(all(rsum < 175000000)) %Runs this program until all the pavement budgets ar
    for i = 1:length(rsum)
        while rsum(i) > 175000000 %If a certain year is over the pavement budget, thi
            if rsum(i+1) < 175000000 %Checks if the next year's budget is also overbu
                [val, row] = max(combined_data.(i+15)(1511:height(combined_data))); %
                row = row + 1510;
                if not(strcmp(combined_data.(i+2){row}(1:2),'**')) %Ensures this proj
                    combined_data.(i+16)(row) = val;
                    combined_data.(i+15)(row) = NaN;
                    combined_data.(i+3)(row) = combined_data.(i+2)(row);
                    combined_data.(i+3){row} = strcat('*',combined_data.(i+3){row});
                    combined_data.(i+2)(row) = {''};
                    count(1) = count(1) + 1;
```

If the job has already been shifted two years, it will be assigned a placeholder cost of negative infinity so the program can search for the next-most-expensive project.

```
            else %If the project has already been moved two years, then it finds the nex
                tempcost = [];
                ogrow = [];
                while strcmp(combined_data.(i+2){row}(1:2),'**') %This loop sets project
                    tempcost(length(tempcost)+1) = combined_data.(i+15)(row); %#ok<SAGRO
                    ogrow(length(ogrow)+1) = row; %#ok<SAGROW>
                    combined_data.(i+15)(row) = -Inf;
                    [val, row] = max(combined_data.(i+15)(1511:height(combined_data)));
                    row = row + 1510;
                end %Then rplace the temporary cost with its orginal cost
                combined_data.(i+16)(row) = val;
                combined_data.(i+15)(row) = NaN;
                combined_data.(i+3)(row) = combined_data.(i+2)(row);
                combined_data.(i+3){row} = strcat('*',combined_data.(i+3){row});
                combined_data.(i+2)(row) = {''};
                count(1) = count(1) + 1;
                for j = 1:length(ogrow)
                    combined_data.(i+15)(ogrow(j)) = tempcost(j);
                end
            end
```

If both the current year and the following year are over budget, then the program will attempt this process on the second year following the current year.

```
            elseif rsum(i+2) < 175000000 %If years i and i+1 were over budget, program
                [val, row] = max(combined_data.(i+15)(1511:height(combined_data)));
                row = row + 1510;
                if  not(strcmp(combined_data.(i+2){row}(1),'*')) %Again, it must ident
                    combined_data.(i+17)(row) = val; %
                    combined_data.(i+15)(row) = NaN;
                    combined_data.(i+4)(row) = combined_data.(i+2)(row);%
                    combined_data.(i+4){row} = strcat('**',combined_data.(i+4){row});
                    combined_data.(i+2)(row) = {''};
                    count(2) = count(2) + 1;
```

If all of the current year, the following year, and the second year are over budget, then the program will move the current scanning year to the following year, then return to it later.

```
            else
                i = i+1;
            end
```

The same process is used on single bridge jobs in the file *shiftsinglebridges.m*, except they can be moved up to five years instead of two, and their constraint is the $250M total budget (so rather than summing rows 1 to 1510, the entire column sum is used).

```
for i = 1:12
    colsum(i) = nansum(combined_data.(i+15));
end
```

6. Identifying Project Years

Now that the budget is met for each year, the program next identifies where jobs can be packaged together. The script *CombinedDirectMatch.m* builds off the “possible” tab created with SQL to search for job groupings in each year. Again, the details are commented in the code, but the general logic is as follows.

The program goes down each row of “possible” adding the keys to a cell array. It then takes this cell array and scans the *combined_data* spreadsheet, adding job names to smaller cell arrays called “work20”, “work21”, etc. This merges projects from the bridges and pavement segments with shared locations into a single cell array for each year.

```
for i = 1:length(possible.TargetID) %Adds each possible key to
    key{1} = possible.CKey1(i);
    key{2} = possible.CKey2(i);
    key{3} = possible.CKey3(i);
    key{4} = possible.CKey4(i);
    key{5} = possible.CKey5(i);
    for j = 1:length(combined_data.CKey) %Scans possible key fr
        for k = 1:length(key)
            if strcmp(key{k},combined_data.CKey(j)) %Adds work
                work20{k} = combined_data.Work_Done_in_2020{j};
                work21{k} = combined_data.Work_Done_in_2021{j};
                work22{k} = combined_data.Work_Done_in_2022{j};
                work23{k} = combined_data.Work_Done_in_2023{j};
                work24{k} = combined_data.Work_Done_in_2024{j};
                work25{k} = combined_data.Work_Done_in_2025{j};
                work26{k} = combined_data.Work_Done_in_2026{j};
                work27{k} = combined_data.Work_Done_in_2027{j};
                work28{k} = combined_data.Work_Done_in_2028{j};
                work29{k} = combined_data.Work_Done_in_2029{j};
                work30{k} = combined_data.Work_Done_in_2030{j};
                work31{k} = combined_data.Work_Done_in_2031{j};
            end
        end
```

Here is an example of an output: The program finished with row 802, where Bridge 38022 and Pavement Segment R12514 share a common TargetID, meaning their jobs can be packaged into a project if they have jobs in the same year.

802x7 table

	1 TargetID	2 count	3 CKey1	4 CKey2	5 CKey3	6 CKey4	7 CKey5
789	'66400202...	'2'	'R12225'	'37956'			
790	'66400202...	'2'	'37957'	'R12229'	''	''	''
791	'66400300...	'2'	'37959'	'R12231'	''	''	''
792	'66400300...	'2'	'46076'	'R12233'	''	''	''
793	'66400301...	'2'	'37961'	'R12237'	''	''	''
794	'66400500...	'2'	'R12252'	'37964'	''	''	''
795	'66400800...	'2'	'R12264'	'37967'	''	''	''
796	'66400801...	'2'	'R12269'	'37968'	''	''	''
797	'66400902...	'2'	'37971'	'R12295'	''	''	''
798	'66401100...	'2'	'R12304'	'37973'	''	''	''
799	'66401900...	'2'	'37985'	'R12352'	''	''	''
800	'66402800...	'2'	'R12414'	'42233'	''	''	''
801	'66404001...	'2'	'R12498'	'38015'	''	''	''
802	'66404500...	'2'	'38022'	'R12514'	''	''	''

In "work30" there is a job in column 2, meaning that R12514 has a job scheduled for 2030, but 38022 does not because column 1 is empty.

results | work30 | possible | work27 | wo

1x2 cell

	1	2	3	4	5
1		Mechaniz...			
2					
3					
4					

The program takes "work" arrays for each year and searches for instances where the array has more than one job, meaning it has found a project package.

```
%20
matches = 0;
%Looks for non blanks in each year,
for j = 1:length(work20)
    if work20{j} ~= ""
        matches = matches + 1;
    end
end
if matches > 1
    results(i).Match2020 = "Yes";
else
    results(i).Match2020 = "No";
end
```

This cycle repeats for each year up to 2031. After scanning each row, the program adds a result of "Yes" or "No" to a results table with this format, also building off the "possible" table created in SQL.

Fields	CKey1	CKey2	CKey3	CKey4	CKey5	Match2020	Match2021	Match2022	Match2023	Match2024	Match2025	Matc
42	'122'	'R338'	''	''	''	"No"	"No"	"No"	"No"	"No"	"No"	"No"
43	'R341'	'124'	''	''	''	"No"	"No"	"No"	"No"	"No"	"No"	"No"
44	'R347'	'125'	''	''	''	"No"	"No"	"No"	"No"	"No"	"No"	"No"
45	'R398'	'47857'	'142'	''	''	"No"	"No"	"No"	"Yes"	"No"	"No"	"No"
46	'147'	'R409'	''	''	''	"No"	"No"	"No"	"No"	"No"	"No"	"Yes"
47	'149'	'R411'	''	''	''	"No"	"No"	"No"	"No"	"Yes"	"No"	"No"
48	'150'	'R414'	''	''	''	"No"	"No"	"No"	"Yes"	"No"	"No"	"No"
49	'151'	'R415'	''	''	''	"No"	"No"	"No"	"No"	"No"	"Yes"	"No"
50	'152'	'R417'	''	''	''	"No"	"No"	"No"	"No"	"No"	"No"	"No"
51	'155'	'R429'	'156'	''	''	"No"	"No"	"No"	"Yes"	"Yes"	"No"	"No"
52	'157'	'R433'	''	''	''	"No"	"No"	"No"	"No"	"No"	"No"	"No"
53	'45818'	'R436'	''	''	''	"No"	"No"	"No"	"No"	"No"	"No"	"No"
54	'164'	'R448'	''	''	''	"No"	"No"	"No"	"No"	"No"	"No"	"No"
55	'167'	'168'	''	''	''	"No"	"No"	"No"	"No"	"No"	"No"	"No"
56	'178'	'179'	''	''	''	"No"	"No"	"No"	"No"	"No"	"No"	"No"
57	'192'	'R579'	''	''	''	"No"	"No"	"No"	"No"	"No"	"No"	"No"
58	'193'	'R581'	''	''	''	"No"	"No"	"No"	"No"	"No"	"No"	"No"

7. Assembling and Displaying Projects

Now that the years of projects have been identified, they need to be displayed in a readable output that shows project location, type, cost, and year. The script *usedkeys.m* completes this output. This script uses the "results" table to search for years where project packaging is possible. When the program arrives at a value of "Yes," it runs a user-created function called *findpairs.m* to extract packages from the master spreadsheet, *combined_data*.

```
    if strcmp(results.(ctyear)(i),"Yes") %If the program sees "Yes" in the results table, it has identified the location and ye
        [Key1, Key2, Key3, Key4, Key5, Cost1, Cost2, Cost3, Cost4, Cost5, Proj1, Proj2, Proj3, Proj4, Proj5] = findpairs(i,j);
        keys = {Key1, Key2, Key3, Key4, Key5};
        cost = [Cost1, Cost2, Cost3, Cost4, Cost5];
        proj = {Proj1, Proj2, Proj3, Proj4, Proj5};
```

The *findpairs* function works as follows.

```
[Key1, Key2, Key3, Key4, Key5, Cost1, Cost2, Cost3, Cost4, Cost5, Proj1, Proj2, Proj3, Proj4, Proj5] = findpairs(row,year)
```

The function has two inputs: "row" and "year", and 15 outputs: Keys 1-5, Costs 1-5, and Project 1-5. The function initiates a cell array with all the "CKeys" from each row and year of results where *usedkeys.m* finds a "Yes" output. It also creates two 1x5 blank cell arrays for each of these three outputs to fill.

```
key = cell(1,5); %Create a cell array for t
key{1} = results.CKey1(row);
key{2} = results.CKey2(row);
key{3} = results.CKey3(row);
key{4} = results.CKey4(row);
key{5} = results.CKey5(row);

cost = zeros(1,5); %Create a vector for the
stryear = num2str(year);
ctyear = strcat("Cost_",stryear);

proj = cell(1,4); %Create a cell array for
ctyear2 = strcat("Work_Done_in_",stryear);
```

With the key cell array filled in, the function now extracts costs and project names from *combined_data.*

```
for i = 1:height(combined_data) %The program v
    if strcmp(combined_data.CKey{i},key{1})
        cost(1) = combined_data.(ctyear)(i);
        proj{1} = combined_data.(ctyear2)(i);
    end
    if strcmp(combined_data.CKey(i),key{2})
        cost(2) = combined_data.(ctyear)(i);
        proj{2} = combined_data.(ctyear2)(i);
    end
    if strcmp(combined_data.CKey(i),key{3})
        cost(3) = combined_data.(ctyear)(i);
        proj{3} = combined_data.(ctyear2)(i);
    end
    if strcmp(combined_data.CKey(i),key{4})
        cost(4) = combined_data.(ctyear)(i);
        proj{4} = combined_data.(ctyear2)(i);
    end
    if strcmp(combined_data.CKey(i),key{5})
        cost(5) = combined_data.(ctyear)(i);
        proj{5} = combined_data.(ctyear2)(i);
    end
end
```

To avoid errors in the program, a block of code is added to remove any bugged project names that have no associated costs.

```
for i = 1:length(cost) %Any project wi
    if isnan(cost(i)) || cost(i) == 0
        key{i}{1} = '';
        proj{i}{1} = '';
    end
end
```

The keys, costs, and project names are finally published as the output variables of the function.

```
Key1 = key{1}{1};
Key2 = key{2}{1};
Key3 = key{3}{1};
Key4 = key{4}{1};
Key5 = key{5}{1};
Cost1 = cost(1);
Cost2 = cost(2);
Cost3 = cost(3);
Cost4 = cost(4);
Cost5 = cost(5);
Proj1 = proj{1}{1};
Proj2 = proj{2}{1};
Proj3 = proj{3}{1};
Proj4 = proj{4}{1};
Proj5 = proj{5}{1};
end
```

Now, returning to the *usedkeys.m* script, the outputs of this function are put into three cell arrays.

```
keys = {Key1, Key2, Key3, Key4, Key5};
cost = [Cost1, Cost2, Cost3, Cost4, Cost5];
proj = {Proj1, Proj2, Proj3, Proj4, Proj5};
```

These results are finally published into a table called "usedkeys" for easy viewing of project locations, costs, years, and names.

```
for k = 1:length(keys)
    usedkeys(length(usedkeys)+1).Key = keys{k};
    usedkeys(length(usedkeys)).Year = j;
    usedkeys(length(usedkeys)).Cost = cost(k);
    usedkeys(length(usedkeys)).Project = proj{k};
end
```

	1 Key	2 Year	3 Cost	4 Project
1	'R30'	2028	1708	'Mill, Mec...
2	'13'	2028	7000	'CM_Deck'
3	''	2028	0	''
4	''	2028	0	''
5	''	2028	0	''
6	'R131'	2031	1005	'Mechani...
7	'42'	2031	3000	'CM_Super'
8	''	2031	0	''
9	''	2031	0	''
10	''	2031	0	''
11	'R144'	2024	6715	'Mechani...
12	'49'	2024	2000	'CM_Sub'
13	''	2024	0	''
14	''	2024	0	''
15	''	2024	0	''
16	'R145'	2020	261496	'Microsur...
17	'50'	2020	131000	'Sub_Reh

8. Exporting Results to Excel

A simple stand-alone script can be used to export the results into Microsoft Excel. In this instance, three sheets are being published to the Excel file called *logic1.xlsx*.

```
writetable(combined_data,'logic1.xlsx','Sheet','AfterShifts');
writetable(results,'logic1.xlsx','Sheet','PackageYears');
writetable(usedkeys,'logic1.xlsx','Sheet','Packages');
```

Simple Excel formulas can be used in this spreadsheet to display final budget results and project packages in an organized manner.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	
1	CKey	TargetID	Work_Done_in_2020	Work_Done_in_2021	Work_Done_in_2022	Work_Done_in_2023	Work_Done_in_2024	Work_Done_in_2025	Work_Done_in_2026	Work_Done_in_2027	Work_Done_in_2028	Work_Done_in_2029	Work_Done_in_2030	Work_Done_in_2031	[redacted]	Cost_2020	Cost_
2	2	0100150030		Epx								Epx					
3	3	0100150031			Struct_Overlay					CM_Super							
4	4	0100150050		Epx								Epx					
5	5	0100150051								CM_Deck	CM_Super	CM_Sub					
6	12	0100150150	Epx									Epx				168000	
7	13	0100150151	CM_Deck								CM_Deck					6000	
8	14	0100150190								CM_Sub			CM_Deck				
9	15	0100150191											CM_Sub				
10	18	0100150230		Epx								Epx					
11	19	0100150231	CM_Super				CM_Sub				CM_Super					5000	
12	24	0100150330												CM_Super			
13	25	0100150331						CM_Deck	CM_Super	CM_Sub							
14	27	0100150390							CM_Super			CM_Sub					
15	28	0100150391	CM_Deck	CM_Sub					CM_Super		CM_Deck	CM_Sub				4000	
16	33	0100150450						CM_Deck	CM_Super					CM_Sub			
17	34	0100150480	Struct_Overlay									CM_Sub				779000	
18	36	0100160030											CM_Sub				
19	38	0100160080	Sub_Rehab										CM_Super			50000	
20	39	0100160090	Full_Paint					Sub_Rehab								201000	
21	40	0100300030	CM_Deck	CM_Super	CM_Sub			Culv_Rep_Box								2000	
22	41	0100300110	CM_Deck	CM_Super	CM_Sub	Culv_Rep_Box										2000	
23	42	0100300200									CM_Deck			CM_Super			
24	43	0100300210				Epx					CM_Super						
25	48	0100300300			CM_Super	CM_Sub											
26	49	0100300310			CM_Super		CM_Sub										
27	50	0100300320	Sub_Rehab										CM_Deck	CM_Super		131000	
28	52	0100300410	CM_Deck	CM_Super	CM_Sub						CM_Deck	CM_Super	CM_Sub			3000	
29	53	0100300420		CM_Deck	CM_Super	CM_Sub											
30	54	0100300450	CM_Super	CM_Sub			Sup_Rpl									6000	
31	59	0100340060				CM_Deck	CM_Super	CM_Sub									
32	61	0100340110	CM_Deck	CM_Super	CM_Sub						CM_Deck	CM_Super	CM_Sub			3000	
33	63	0100340110							CM_Sub								
34	64	0100340120					CM_Sub										
35	69	0100340190	CM_Super		CM_Deck	CM_Sub								CM_Super		2000	
36	72	0100340240							Culv_Rep_Box								
37	74	0100340260				CM_Deck	CM_Super	CM_Sub									
38	76	0100940120		Struct_Overlay													
39	77	0100940130						CM_Sub	CM_Super	CM_Deck							
40	80	0100940240							CM_Sub								
41	84	0100970010						Culv_Rep_Box									
42	85	0100970010									Culv_Rep_Box						
43	86	0100970040	CM_Deck	CM_Super		CM_Sub								CM_Deck		2000	
44	88	0100970120	CM_Deck	CM_Super		CM_Sub								CM_Deck		2000	
45	93	0101160010									CM_Deck						
46	97	0101160090	CM_Super	CM_Sub							CM_Super		CM_Sub			3000	
47	99	0101160150										Sup_Rpl					

BeforeShifts | AfterShifts | PossiblePackages | Packages | PackageYears | +

	A	B	C	D	E	F	G
1	Key	Year	Cost	Project			
2	R30	2028	1708	Mill, Mechanized Edge Patch (HMA&COMP)	Packaged Projects	650	
3	13	2028	7000	CM_Deck	Total Projects	16537	
4		2028	0		Percent Packaged	3.93%	
5		2028	0				
6		2028	0				
7	R131	2031	1005	Mechanized Patch (HMA&COMP)			
8	42	2031	3000	CM_Super			
9		2031	0				
10		2031	0				
11		2031	0				
12	R144	2024	6715	Mechanized Patch (HMA&COMP)			
13	49	2024	2000	CM_Sub			
14		2024	0				
15		2024	0				
16		2024	0				
17	R145	2020	261496	Microsurface/Thin Overlay (HMA&COMP)			
18	50	2020	131000	Sub_Rehab			
19		2020	0				
20		2020	0				
21		2020	0				
22	R145	2031	4098	Mechanized Patch (HMA&COMP)			
23	50	2031	13000	CM_Super			
24		2031	0				
25		2031	0				
26		2031	0				
27	52	2028	3000	CM_Deck			
28	R156	2028	1195	Mechanized Patch (HMA&COMP)			
29		2028	0				
30		2028	0				
31		2028	0				
32	R182	2025	1149	Base Repair, Manual Patch (HMA&COMP)			
33	59	2025	2000	CM_Sub			
34		2025	0				
35		2025	0				
36		2025	0				
37		2026					

BeforeShifts | AfterShifts | PossiblePackages | Packages | PackageYears | +

Process 2 (Prioritizing Project Packages First)

As discussed in the overview, the logic of the second process is to shift single jobs to form projects where possible, then shift remaining single jobs to meet budgetary constraints.

1-4. Standardizing Excel Data to Load into MATLAB

Steps 1-4 of the second process follow the same process described in Process 1 where PAMS and BAMS data is standardized, SQL scripts are run to remove extraneous data and identify potential pairings, and the data is loaded into MATLAB as *combined_data.*

5. Identifying Project Years and Assembling Projects Without Delayed Jobs

Step 5 of this process is equivalent to Steps 6-7 of the first process where the scripts *CombinedDirectMatch.m and usedkeysdirect.m* create a results table with a binary "Yes" or "No" output and a "usedkeys" table displaying projects for jobs scheduled in the same year. The key difference in this process is that *usedkeysdirect.m* removes packaged jobs from the master data set *combined_data*. This ensures that repeat jobs are not assembled as the program moves from a zero-year delay down the line to a five-year delay.

```
for k = 1:length(combined_data.CKey)
    if strcmp(combined_data.CKey(k),keys{1}) || strcmp(combined_data.CKey(k),keys{2}) || strcmp(combined_data.CKey(k),keys{3})
        combined_data1.(ctyear2){k} = '';
        combined_data1.(ctyear3)(k) = NaN;
    end
end
```

6. Identifying Project Years and Assembling Projects with Delayed Jobs

Similar scripts are run to identify and assemble projects where a shift is required to match a job in one year with a different job in the same location that occurs in a later year. In this example, pavement jobs can be moved up to three years, and bridge jobs can be moved up to five years.

The example shown below is the script *MatchWithWindow1.m,* identifying projects that require jobs to be shifted one year for assembly to 2021. Notice that instead of looking for jobs in only the cell array "work20", the program looks for blanks existing in both "work20" and "work21." As in Process 1, at least one match will result in an output of "Yes" to the "results" table for the year 2021. This process continues down the line until the year 2031.

```
    %21
    %Looks for non blanks in each year, which means multiple projects
    matches = 0;
    for j = 1:(length(work20)-1)
        if work20{j} ~= "" && (work21{j+1} ~= "" || work21{j+2} ~= "" || work21{j+3} ~= "")
            matches = matches + 1;
        end
        if work21{j} ~= "" && (work20{j+1} ~= "" || work20{j+2} ~= "" || work20{j+3} ~= "")
            matches = matches + 1;
        end
    end
    if matches > 0
        results1(i).Match2021 = "Yes";
    else
        results1(i).Match2021 = "No";
    end
```

The script *usedkeys1year.m* is the equivalent of *usedkeysdirect.m*, except that it assembles projects identified by *MatchWithWindow1.m* instead of *CombinedDirectMatch.m*. The following table shows which scripts are used at what points in the process.

Job Shift (Years of Delay)	**Identifying Project Years**	**Assembling Projects**
0	*CombinedDirectMatch.m*	*usedkeysdirect.m (with findpairs.m function)*
1	*MatchWithWindow1.m*	*usedkeys1year.m (with findpairs.m function)*
2	*MatchWithWindow2.m*	*usedkeys2year.m (with findpairs2.m function)*
3	*MatchWithWindow3.m*	*usedkeys3year.m (with findpairs3.m function)*
4	*MatchWithWindow4.m*	*usedkeys4year.m (with findpairs4.m function)*
5	*MatchWithWindow5.m*	*usedkeys5year.m (with findpairs5.m function)*

Upon reaching the rows requiring three to five years of shifting, the programs must be modified to neglect pavement projects. The image below shows how the program searches for keys that do not begin with "R", which indicates a pavement project.

```
%24
%Looks for non blanks in each year, which means multiple projects
matches = 0;
for j = 1:(length(work20)-1)
    if (not(strcmp(work20{j},"")) && not(strcmp(key{j}{1}(1),'R'))) && (not(strcmp(work24{j+1},"")) || not(strcmp(work24{j+2
        matches = matches + 1;
    end
    if not(strcmp(work24{j},"")) && (((not(strcmp(work20{j+1},""))) && not(strcmp(key{j+1}{1}(1),'R'))) || ((not(strcmp(work
        matches = matches + 1;
    end
end
if matches > 0
    results4(i).Match2024 = "Yes";
else
```

7. Shifting Single Jobs

After projects have been assembled and removed from the master list, the program can shift the remaining stand-alone jobs to meet the budget constraints. The scripts *shiftsinglepavements.m* and *shiftsinglebridges.m* are used in the same m and order as in Process 1, Step 5, with one exception. Because this process forms projects first and then removes them from the master list, it must independently calculate the sum of the project packages, then add their value back into the yearly sum of the single projects. The additional scripts *sumroadproj.m* and *sumtotalproj.m* move through the "usedkeysfinal" table to gather a sum of all yearly costs for pavement projects and total projects, respectively.

The *sumroadproj.m* script searches for packaged jobs with the preceding "R" identifier in the key, then adds the cost to a corresponding element of a vector depending on the scheduled year.

```
for i = 1:12
    rsum(i) = nansum(combined_data.(i+15)(1511:8431));
    rsum(i) = rsum(i) + bundlersum(i); %#ok<SAGROW>
end
```

```
for i = 1:height(usedkeysfinal)
    if strcmp(usedkeysfinal.(1){i}(1),'R')
        if usedkeysfinal.(2)(i) == 2020
            bundlersum(1) = bundlersum(1) + usedkeysfinal.(3)(i);
        end
        if usedkeysfinal.(2)(i) == 2021
            bundlersum(2) = bundlersum(2) + usedkeysfinal.(3)(i);
        end
        if usedkeysfinal.(2)(i) == 2022
            bundlersum(3) = bundlersum(3) + usedkeysfinal.(3)(i);
        end
        if usedkeysfinal.(2)(i) == 2023
            bundlersum(4) = bundlersum(4) + usedkeysfinal.(3)(i);
        end
        if usedkeysfinal.(2)(i) == 2024
            bundlersum(5) = bundlersum(5) + usedkeysfinal.(3)(i);
```

This vector is then added to the sum of single pavement projects in *shiftsinglepavements.m* to acquire the final costs used to meet budget constraints. The same cycle occurs in *shiftsinglebridges.m,* but using the vector acquired in *sumtotalproj.m*.

```
for i = 1:height(usedkeysfinal)
        if usedkeysfinal.(2)(i) == 2020
            bundletsum(1) = bundletsum(1) + usedkeysfinal.(3)(i);
        end
        if usedkeysfinal.(2)(i) == 2021
            bundletsum(2) = bundletsum(2) + usedkeysfinal.(3)(i);
        end
        if usedkeysfinal.(2)(i) == 2022
            bundletsum(3) = bundletsum(3) + usedkeysfinal.(3)(i);
        end
        if usedkeysfinal.(2)(i) == 2023
            bundletsum(4) = bundletsum(4) + usedkeysfinal.(3)(i);
        end
        if usedkeysfinal.(2)(i) == 2024
            bundletsum(5) = bundletsum(5) + usedkeysfinal.(3)(i);
        end
```

```
for i = 1:12
    colsum(i) = nansum(combined_data.(i+15)); %#ok<SAGROW>
    colsum(i) = colsum(i) + bundletsum(i); %#ok<SAGROW>
end
```

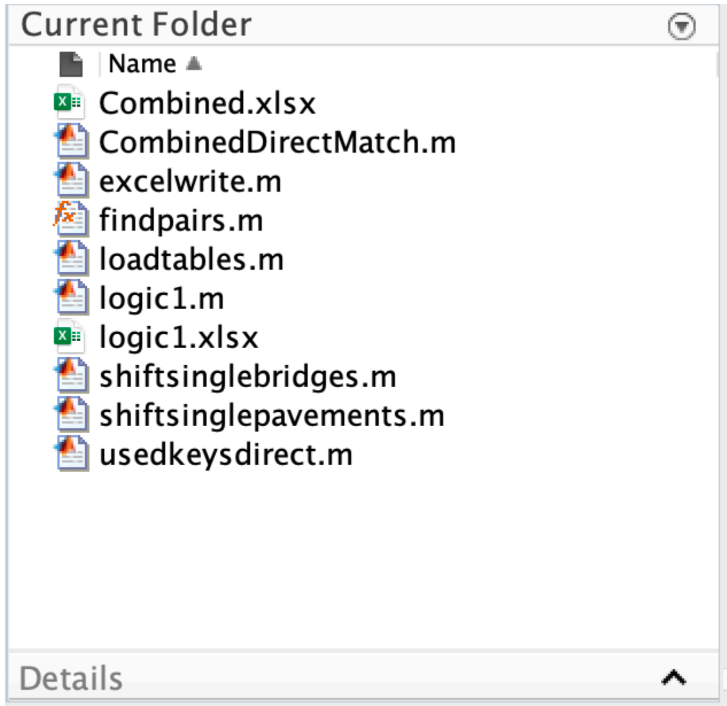
8. Exporting Results to Excel

Finally, the results can be exported into Excel for easy viewing using the same methodology described in Step 8 of Process 1.

Simple Method of Running the Program

Clicking "Run" on several different scripts in the correct order is tedious and prone to error. However, there is a simple way of running the MATLAB scripts all at once after SQL implementation in Step 3 of each process.

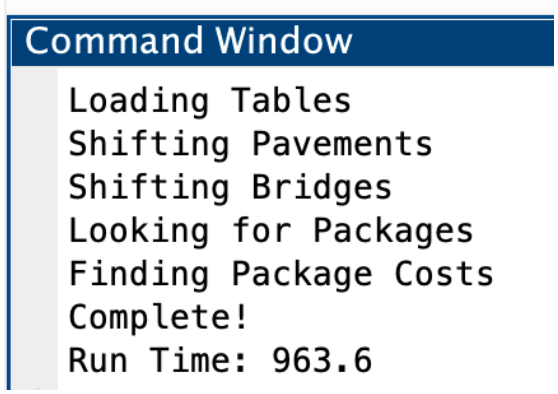
First, the user must make sure all the MATLAB scripts and Excel files required are in the correct and current folder.



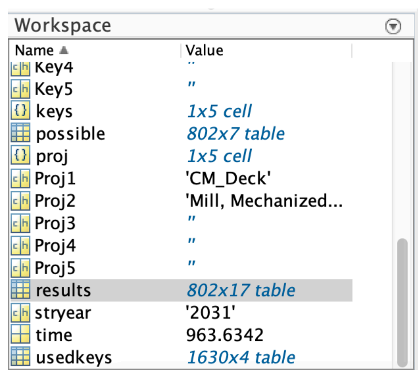
With all the .xlsx and .m files saved in the correct location, a simple executionary script can be written to execute all the scripts in a specific order. In this example, a file called *logic1.m* is used to run Process 1. Messages can also be included to show the user that the program is running. Getting the program started is as simple as clicking "Run."

```
clearvars
clc
tic
disp("Loading Tables");
loadtables;
disp("Shifting Pavements");
shiftsinglepavements;
disp("Shifting Bridges");
shiftsinglebridges;
disp("Looking for Packages");
CombinedDirectMatch;
disp("Finding Package Costs");
usedkeysdirect;
time = toc;
disp("Complete!");
fprintf("Run Time: %.1f\n",time);
```

The command window will display the messages as shown below.



The user may still need to manually run the script to export results to Excel for viewing, or they can include that command in the standalone executionary depending on preference. Typically, it is easier to export manually to adjust sheet and file names with ease. The user can also view the output results without exporting to Excel by simply double-clicking on a variable in the MATLAB Workspace.



Appendix B: Literature Review

Asset management is a process which aims to optimize both the performance and cost-effectiveness of an asset. It relies upon set objectives and quality information as input and outputs a set of decisions [1]. Transportation Asset Management, focusing solely on the infrastructure of a transportation system, allows system owners to keep the transportation assets in good or better condition than they are currently in, and develop a logical capital budgeting plan while containing costs [2]. The principles of asset management have long been used as a mechanism for sustaining highway (pavement) conditions over time while achieving the lowest life cycle cost. More recently asset management principles have been expanded to other asset classes [3]. Software applications exist that enable asset management within one asset class, however, transportation agencies are evaluated on their respective asset classes as a whole. Managing assets across classes (cross-asset management), including allocating available budgets, is a challenging problem [4].

Laumet and Bruun [4] develop an integer optimization program and utilize a linear formulation and a derivative-free optimization approach, which is more realistic than the linearity assumption. In this latter model, the state space of all possible budget distributions among all asset types is explored and the most favorable is selected. This approach, while realistic due to the non-linearity, can be very slow and not practical for a large transportation system [4]. Other optimization models operate in a similar manner by maximizing benefits while considering the effect on adjacent assets [5], formulating the project scheduling as done in maintenance or a bi-level staging problem using dynamic programming to determine fund allocation and project prioritization [6].

Due to the nature of cross-asset allocation, optimal allocation is not straightforward, nor practical at a large-scale sense. Many heuristics have also been developed. One of these, done for the State of Iowa [7], utilizes grouping for assets with similar characteristics and distributes funding to groups based on asset types, then applies a needs-based approach to prioritize assets within each group. Future valuation based on those decisions is predicted and optimized.

Many of these models do not tackle important issues such as the exploitation of interconnected assets and data, incorporation of qualitative and holistic objectives and asset substitution effects—or the consequence on one asset from the failure of a related asset [8]. A handbook for cross-asset allocation for transportation was published as a result of research from an American Association of State Highway and Transportation Officials (AASHTO) grant program [9]. The handbook compiles the output of that research and yields a framework for determining which performance measures to consider across assets. It is a weighted approach not dissimilar to the Analytical Hierarchy Process and also allows for an analysis of the risk of various scenarios. This may be useful at a more macro level but not at the bottom level of allocating among all the individual assets within each class. Another similar framework [10] combines performance measures upwards into more comprehensive measures and uses a ranking scale to determine budget allocations.

The problem of cross-asset allocation is not new but certainly emerging. Techniques do exist for allocating budgets across assets at the higher level where the data set is smaller, however the approaches that consider the lower, individual asset level are cumbersome at best. Optimization-based approaches will yield accurate and useful results but scalability is a limiting factor for implementation within a transportation system due to necessary computing power, programming, and run time. This research aims to bridge the gap by providing a heuristic to prioritize funds at a micro level of transportation decisions.

References

[1] AASHTO Subcommittee on Asset Management in 2004, Jan 2006. [Online]. Available: http://www.transportation.org/sites/scoh/docs/Motion_Trans_Asset_Management.doc.

[2] Federal Highway Administration, "What is Transportation Asset Management?," 17 June 2017. [Online]. Available: https://www.fhwa.dot.gov/asset/if08008/amo_02.cfm. [Accessed 5 May 2021].

[3] U.S. Department of Transportation, Federal Highway Administration, "Executive Brief: Advancing a Transportation Asset Management Approach," 2012.

[4] P. Laumet and B. Mikkel, "Trade-off Analysis for Infrastructure Management: New Approaches to Cross-asset Challenges," *Transportation Research Procedia,* vol. 14, pp. 422-429, 2016.

[5] S. Wang and E. Chou, "Cross-asset transportation project coordination with integer programming and constraint programming," *Transportation Research Record,* vol. 2482, pp. 117-125, 2015.

[6] H. Peng, Z. CHen and L. Sun, "A bilevel program for solving project scheduling problems in network level," *Journal of Tongji University,* vol. 38, pp. 380-385, 2010.

[7] Y. Abukhalil, "Cross asset resource allocation framework for pavement and bridges in Iowa," Iowas State University, 2019.

[8] S. P. A. Petchrompo, "A review of asset management literature on miulti-asset systems," *Reliability Engineering & System Safety,* vol. 181, pp. 181-201, 2019.

[9] Maggiore M and F. KM., "Guide to Cross-Asset Resource Allocation and the Impact on Transportation System Performance," 2015.

[10] M. S. Dehghani, F. Giustozzi, F. G. W. and M. Crispino, "Cross-asset resource allocation framework for achieving performance sustainability," *Transportation research record,* vol. 236, no. 1, pp. 16-24, 2013.

[11] T. S. K. Fwa and J. Riverson, "Highway routine maintenance programming at network level," *Hournal of Transportation Engineering,* vol. 114, pp. 539-554, 1988.